四、IP 核应用之计数器

实验目的：了解 FPGA 的 IP 核相关知识并以计数器 IP 核为例学会基本 IP 使用的流程

实验平台：无

实验原理：

IP 核（Intellectual Property core），也被称为知识产权核，其分为软核、硬核和固核。软核通常是与工艺无关、具有寄存器传输级硬件描述语言描述的设计代码，可以进行后续设计；硬核是前者通过逻辑综合、布局、布线之后的一系列工艺文件，具有特定的工艺形式、物理实现方式；固核则通常介于上面两者之间，它已经通过功能验证、时序分析等过程，设计人员可以以逻辑门级网表的形式获取。

FPGA 的开发方式分为三种，分别是：原理图、Verilog HDL 以及 IP 核。其中原理图方式在较大工程中由于其局限性使用的越来越少，不推荐再学习；Verilog HDL 为当今主流的设计方式；用 IP 核代替用户自己设计的逻辑，可以大大缩短开发周期，提供更加有效的逻辑综合和实现。

Altera IP 核既包括了诸如逻辑和算术运算等简单的 IP 核，也包括了诸如数字信号处理器、以太网 MAC、PCI/PCIE 接口比较复杂的系统模块。Quartus II 中使用 Mega Wizard 插件管理器可以用以修改和创建包含定制 IP 核的设计文件，然后在设计文件中例化 IP 核。在 Mega Wizard 插件管理器中可以创建、定制和例化 Altera IP 核、参数化模型库（LPM）模块以及在 Quartus II 软件、EDA 设计入口和综合工具使用的 IP 核。

实验内容：

新建一个以名为 counter_ip 的工程保存在 prj 下，然后单击 Tools—Mega Wizard Plug-In Manager 来启动 Mega Wizard 插件管理器 。

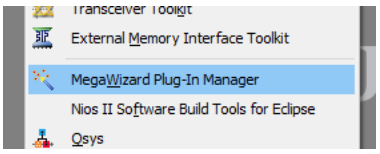

图 5-1 启动 Mega Wizard 插件管理器

弹出图 5-2 对话框，这里有三个选项分别是新建一个定制 IP 核、编辑现有的 IP 核以及复制一个现有的定制 IP 核。这里先单击创建一个新的定制 IP。

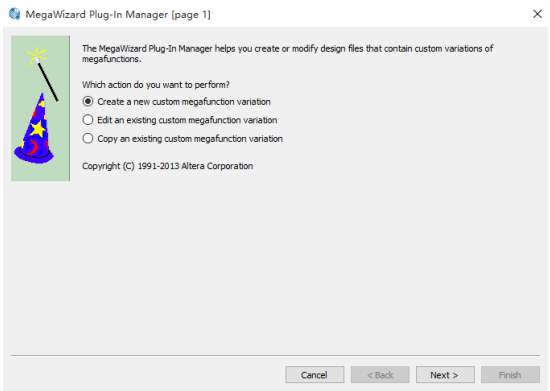

图 5-3 Mega Wizard 插件管理器启动界面

在弹出的图 5-4 Mega Wizard 插件管理器的参数设置界面中，在搜索框中输入 counter 即可显示出 LPM_COUNTER 选中，并将输出目录确定为工程文件夹下的 ip 文件夹，并以 counter 保存，单击 Next。

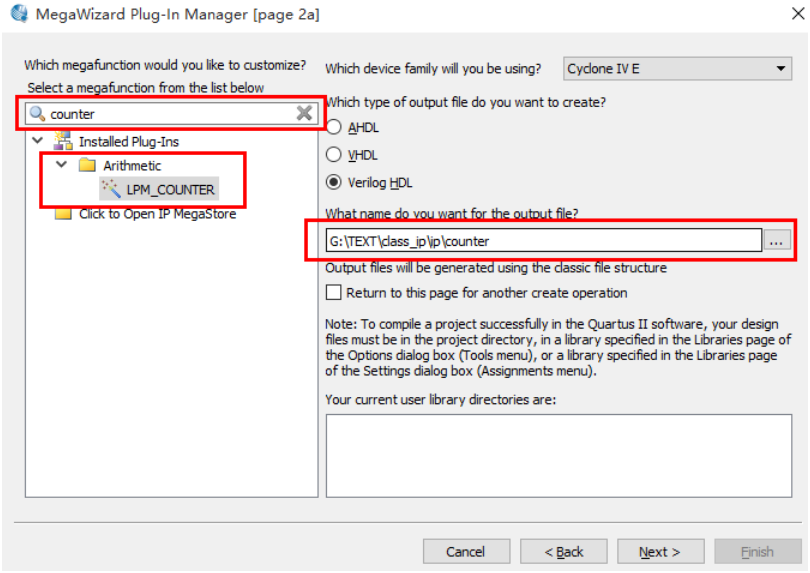

图 5-4 参数设置界面第二页

在弹出的图 5-5 将计数器位数选为 4 位、技术方式为递增计数，这里可以最大支持 256 位输出，单击 Next。

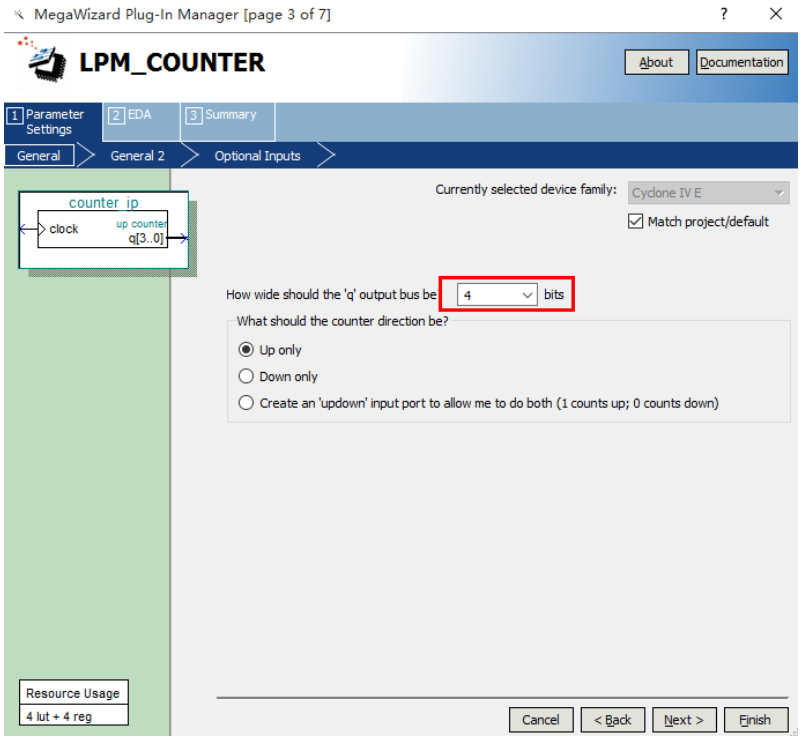

图 5-5 参数设置界面第三页

在弹出的图 5-6 中，发现可选计数使能，时钟使能输入以及支持可选进位输入和进位输出，二进制计数（从 0 开始递增或者从 255 开始递减）和计数值（计数器递增计数到用户指定的计数值或者从用户指定的计数值开始递减计数并重复）。这里先配置为计数值计数且为 10，有进位输入以及输出，单击 Next。

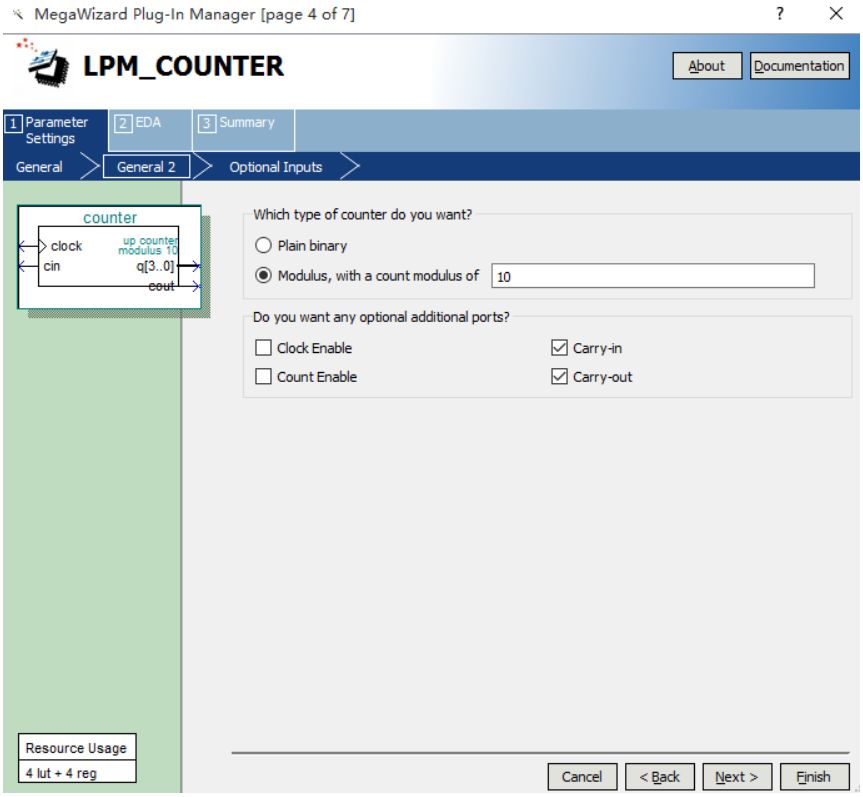

图 5-6 参数设置界面第四页

在弹出的图 5-7 中，发现本 IP 支持可选同步（异步）复位、加载和设置输入。这里先不进行选择，直接单击 Next。

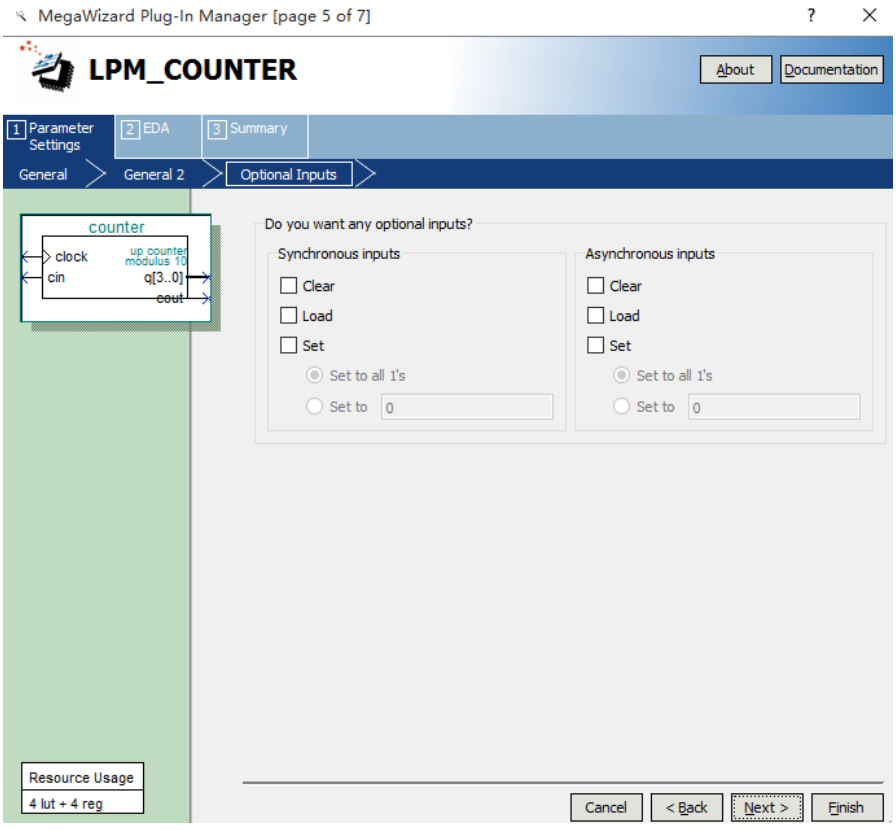

图 5-7 参数设置界面第五页

在弹出的图 5-8 生成的仿真库，使用默认设置直接单击 Next。

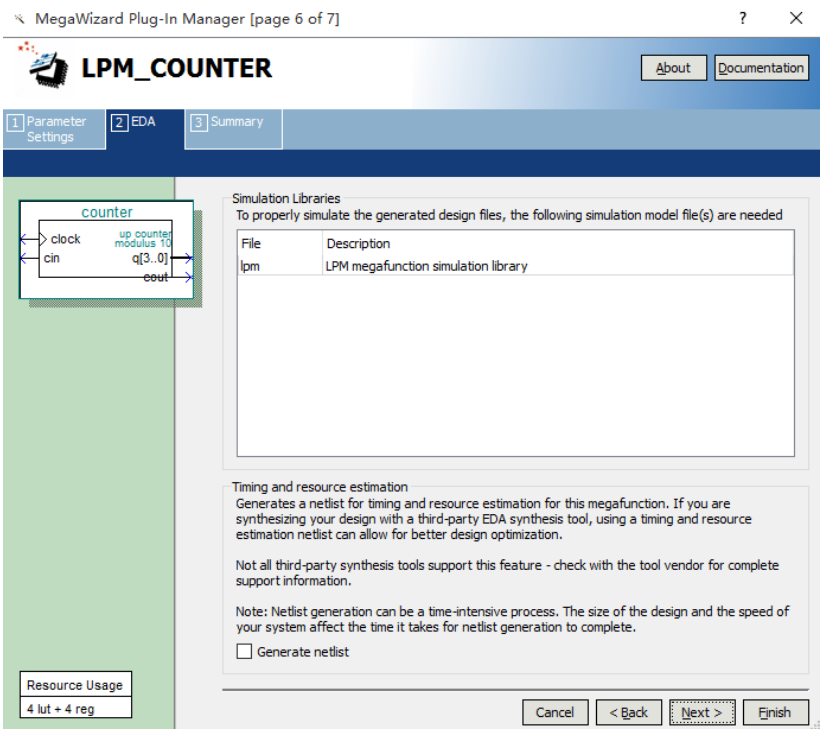

图 5-8 生成仿真库

在弹出的图 5-9 中汇总了可选生成的文件。其中第一个是 VHDL 源文件，对于本设计是必选的；AHDL 包含文件，可以在文本设计文件(.tdf)中使用；VHDL 组件声明文件，可以在 VHDL 设计文件中使用；Quartus II 符号文件可以在原理图设计中使用；例化模板文件以及 VerilogHDL 黑盒文件供给第三方综合工具使用。这里仅勾选黑盒文件点击 Finish。

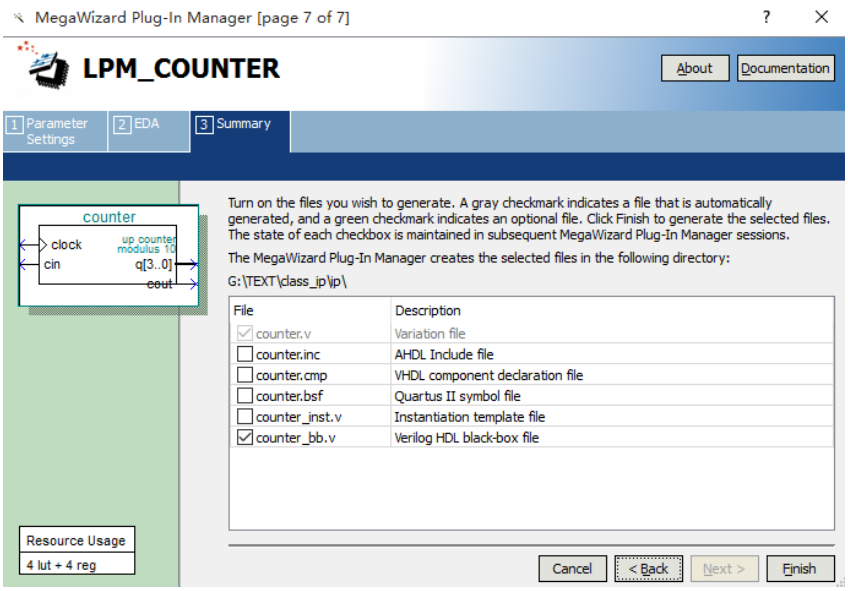

图 5-9 可生成文件汇总界面

在弹出的图 5-10 中提示是否将生成的 IP 核加入工程，这里单击 Yes，即可在工程下看到加入的 IP 核文件。

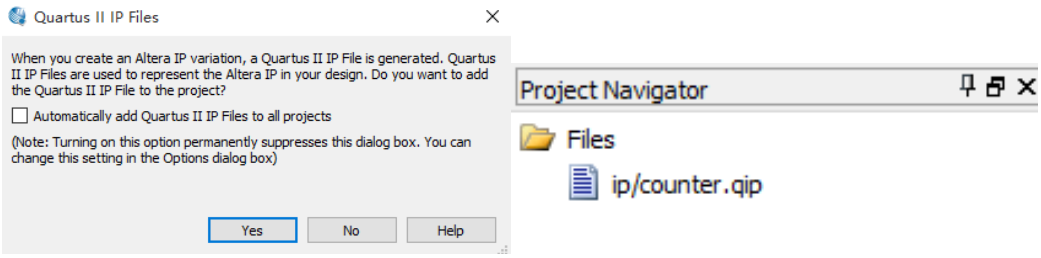

图 5-10 生成的 IP 核加入工程

打开 counter.qip 可以看到只有简单的描述，我们现将生成的 counter.v 添加到工程中。在 Files 右键选择 Add/Remove Files in project，在弹出的对话框选中要添加的文件单击 Add 即将其加入到工程中，点击 Apply 后，单击关闭 Close。

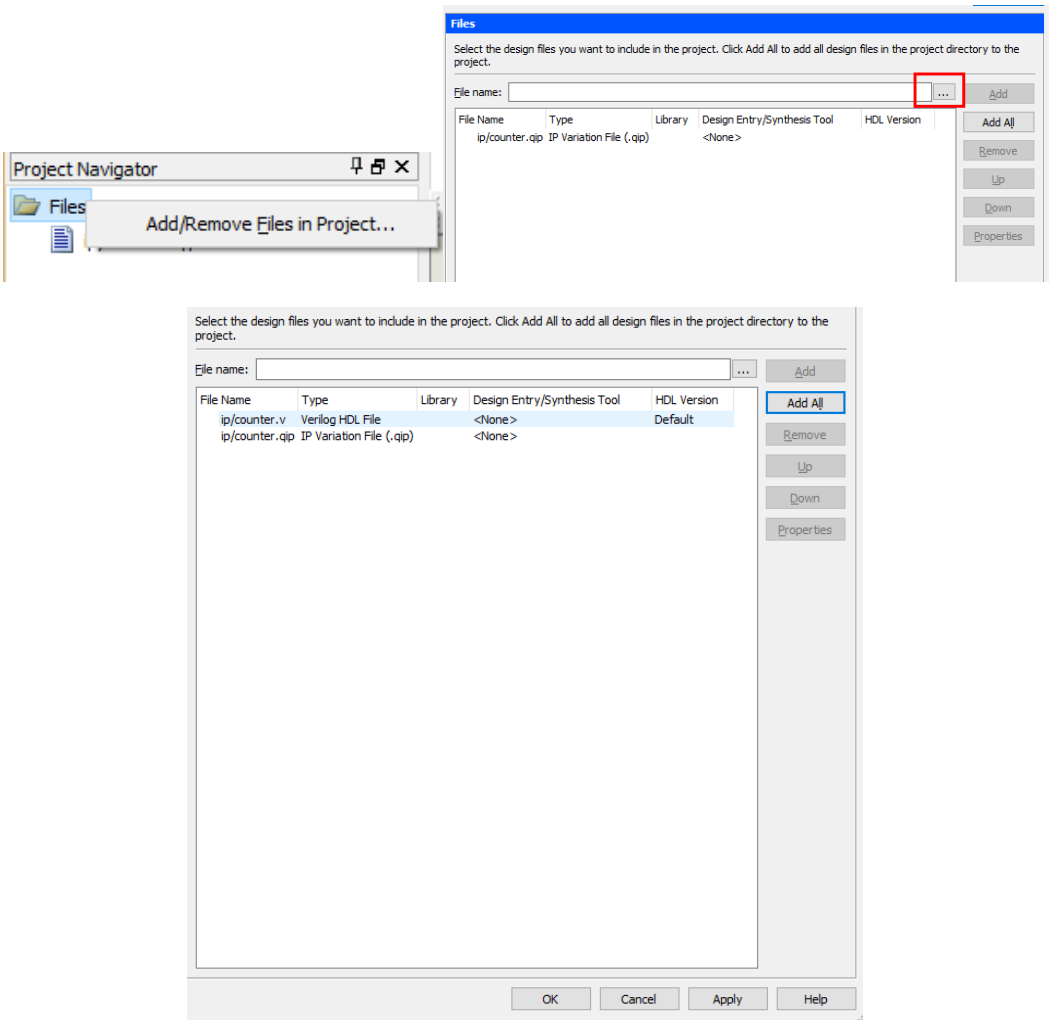

图 5-11 向工程中添加文件

此时可看到在 Files 中已经将其添加进来。进行分析和综合直至没有错误以及警告。为了测试仿真编写测试激励文件，新建 counter_tb.v 文件并输入以下内容再次进行分析和综合直至没有错误以及警告，保存到 testbench 文件夹下。

```
`timescale 1ns/1ns

`define clock_period 20

module counter_tb;
```

```
    reg cin;    //进位输入
    reg clk;    //计数基准时钟

    wire cout;//进位输出
    wire [3:0] q;

    counter counter0(
        .cin(cin),
        .clock(clk),
        .cout(cout),
        .q(q)
    );

    initial clk = 1;
    always #(`clock_period/2)clk = ~clk;

    initial begin
        repeat(20)begin
            cin = 0;
            #(`clock_period*5)cin = 1;
            #(`clock_period)cin = 0;
        end
        #(`clock_period*200);
        $stop;
    end

endmodule
```

设置好仿真脚本后进行功能仿真，可以看到如图 5-12 所示的波形文件。将 q 的显示切换为无符号十进制，可以看出每当 cin 高电平一次，输出 q 进行自加一次直至加满到 9（从 0 开始加）自动清零并开始下一轮计数。符合预期设计，对于此时功能出现毛刺，可先不深究。

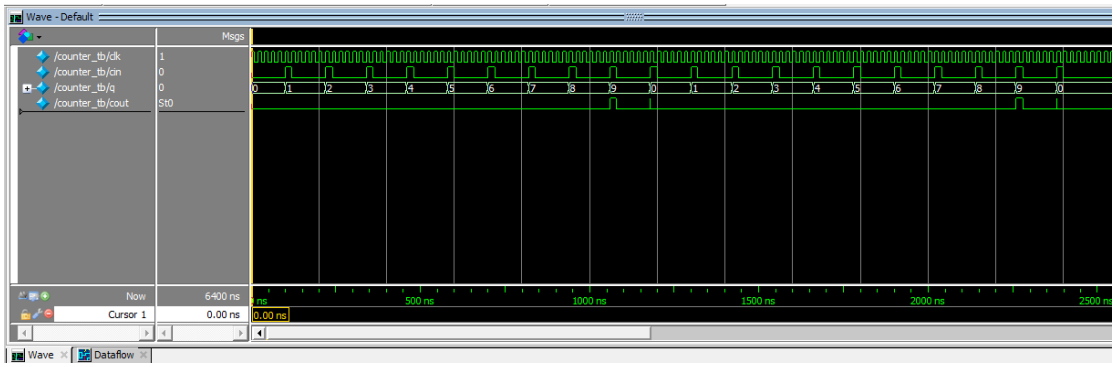

图 5-12 计数值为 10 的功能仿真

现在将 IP 核位数进行更改为二进制计数，打开 Mega Wizard 插件管理器，选择第二项

编辑现有的 IP 核，并选择先前生成的 counter.v。

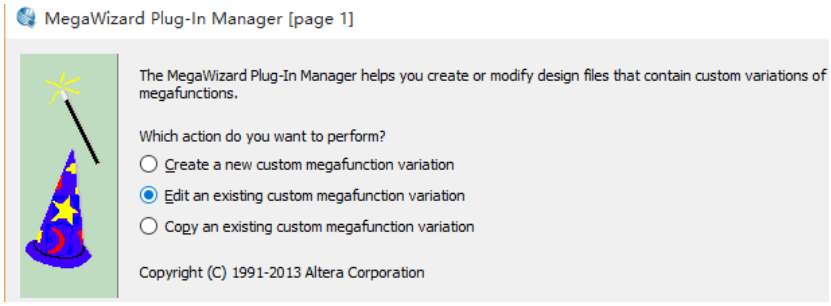

图 5-13-1 编辑现有的 IP 核

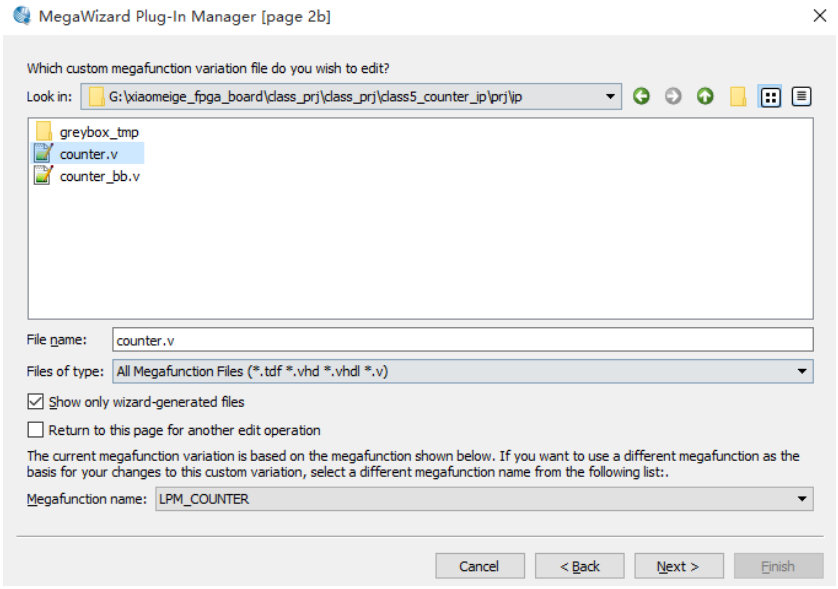

图 5-13-2 编辑现有的 IP 核

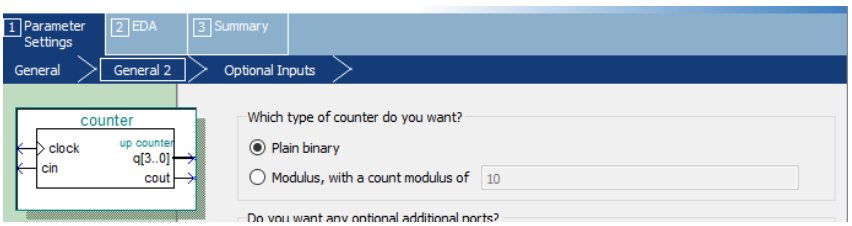

图 5-13-3 编辑现有的 IP 核

重新分析和综合后，在进行功能仿真。出现如图 5-14 的波形文件，可以看到输出 q 在输入 cin 的控制下进行自加一次直至加满到 15（4 位计数器）后自动清零并开始下一轮计数。

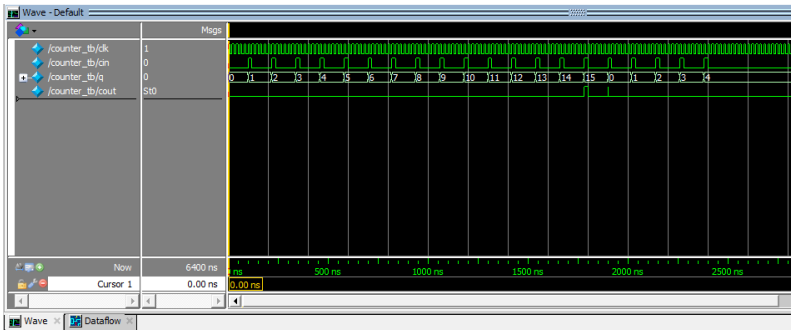

图 5-14 二进制计数功能仿真波形

假设现在想修改设计为 8 位计数器，当然可以再次修改 IP 核设置，此外还可以将两个 4bi 进行级联，即前一级的进位输出连接到下一级的输入端即可，最终将两个分输出拼接为总的输出。

在本工程目录的 rtl 文件夹下新建 verilog file 文件在此文件下输入以下内容并以 counter_top.v 保存，并设置为顶层文件。这里实现了例化设计好的计数器 IP 核，且将其进行两级级联。

```
module counter_top(cin,clk,cout,q);

    input cin;
    input clk;

    output cout;
    output [7:0]q;

    wire cout0;

    counter counter0(
        .cin(cin),
        .clock(clk),
        .cout(cout0),
        .q(q[3:0])
    );

    counter counter1(
        .cin(cout0),
        .clock(clk),
        .cout(cout),
        .q(q[7:4])
    );

endmodule
```

分析综合无误后点击 RTL viewer 可以看如图 5-15 所示的原理图。

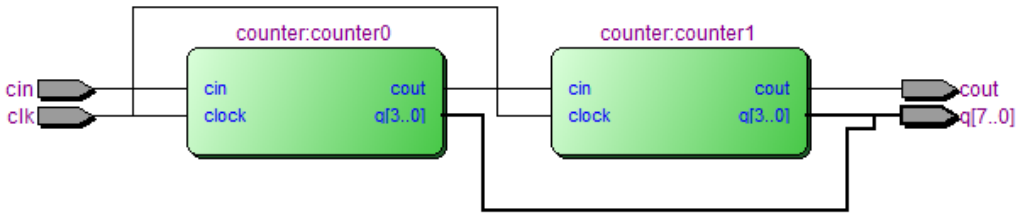

图 5-15 两级级联的计数器原理图

为了测试仿真编写测试激励文件，新建 counter_top_tb.v 文件并输入以下内容再次进行分析和综合直至没有错误以及警告，保存到 testbench 文件夹下。这里除产生了系统时钟以

及复位信号还生成了占空比为 1：5 的 cin 信号。

```
`timescale 1ns/1ns

`define clock_period 20

module counter_top_tb;

    reg cin;    //进位输入
    reg clk;    //计数基准时钟

    wire cout;//进位输出
    wire [7:0] q;

    counter_top counter0(
        .cin(cin),
        .clk(clk),
        .cout(cout),
        .q(q)
    );

    initial clk = 1;
    always #(`clock_period/2)clk = ~clk;

    initial begin
        repeat(300)begin
            cin = 0;
            #(`clock_period*5)cin = 1;
            #(`clock_period)cin = 0;
        end
        #(`clock_period*200);
        $stop;
    end

endmodule
```

设置好仿真脚本后进行功能仿真，可以看到如图 5-16 所示的波形文件，可以看出输出 q 在输入 cin 的控制下进行自加一次直至加满到 255（8 位计数器）后自动清零且产生一个系统时钟周期的 cout 高电平并开始下一轮计数。这里在 cout 高电平后，出现了一个毛刺信号，这里为 IP 核本身问题，此处不做讨论。

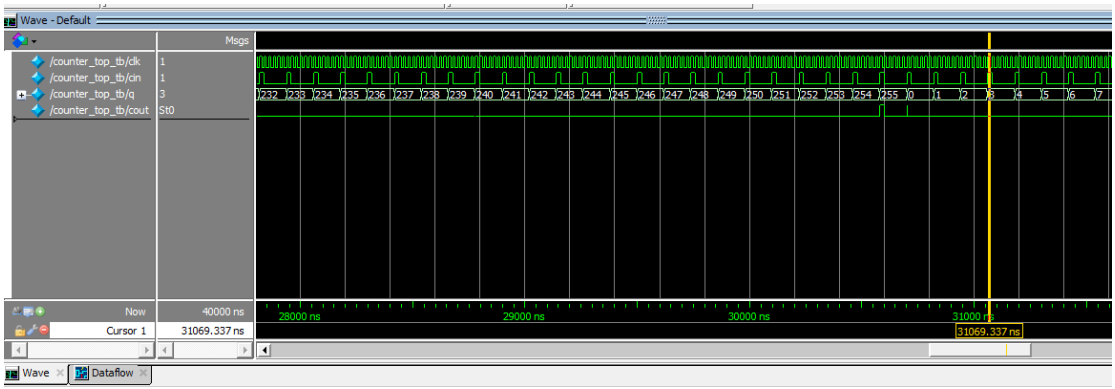

图 5-16 两级级联 4 位计数器功能仿真波形图

现在再次将计数值改为 10 后再仿真，发现功能仿真波形图如图 5-17 所示。可以看出输出 q 在输入 cin 的控制下进行自加一次直至加满到 153 后自动清零并开始下一轮计数。这里之所以是 153，是因为我们将每一级计数器设置为 10，由于计数器从 0 开始加实际计数满值为 9 也就是二进制的 1001，拼接起来就是 1001_1001，即为十进制的 153。这部分的具体应用将在后面课程数码管的使用一讲中详细阐述。

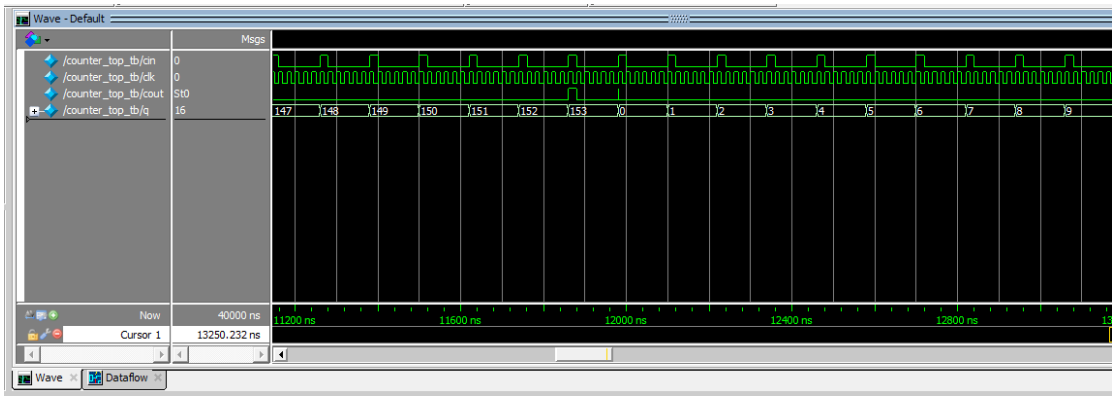

图 5-17 两级级联计数值为 10 的计数器功能仿真波形图

至此就完成了一个基本的基本 IP 使用的流程。请以此为基础自行设计使用其他 IP 核并进行仿真以及板级验证。

如有更多问题，欢迎加入芯航线 FPGA 技术支持群交流学习：472607506

小梅哥
芯航线电子工作室